API User Guide

NOTIFICATIONS AND APPOINTMENT REMINDERS

General Overview **3**

Send a notification & receive replies **4**

First, let's send the message... 4

Next, let's get replies... 8

Receive Replies (in Real-Time) *9*

Receive Replies (On-Demand) *12*

The response you will receive... ..... *17*

Send a notification without replies **20**

So, let's send the message... 21

General Overview

SMS messages are a common and valuable way to send individual notifications and reminder messages, such as appointment reminders. Sending out reminder & notification messages via SMS Central's API is easy.

There's only 1 simple decision to make, before you get started. Do you want to receive response messages, or not?

If you do, then you'll need to be able to receive these messages via the API too.

This tutorial utilises the SMS Central API, please read the API Reference document to help you understand what we're on about in this tutorial.

Send a notification & receive replies

FIRST, LET'S SEND THE MESSAGE...

The first thing you'll need to do is send the message, via a HTTP POST to the SMS Central API URL. Since this is just one single message, here are the required parameters (as described by the API Reference) that will apply:

USERNAME
This is your SMS Central username

PASSWORD
This is your SMS Central password

ACTION
As you are sending a message, the value for this parameter will always be 'send'

ORIGINATOR
This is the number you're sending the message from. If you have a dedicated number, then you would have that number as the value, otherwise, you would use the string value 'shared', to send the message from a pool of shared numbers provided by SMS Central.

RECIPIENT
This is the number (in international format, i.e. 61420314421 for an Australian mobile) that you want to send the message to.

REFERENCE
You can supply a unique (must be unique) reference value, which is relevant to you, and you'll get the same reference value with any reply (so that you can match the reply to the original sent message).

MESSAGE_TEXT
You'd probably have guessed this one, this is the parameter that contains the text of your message. Remember, a message is 160 characters in length, any longer than that and it would be sent as 2 messages, or 3 messages, etc.

Some code

Now that you know what you'd need to send, let's delve into the code itself and how you would do it.

We are providing a PHP example here below, however this can be done with any programming language (we'll add more and more code samples with different languages over time, or let us know if you need some help!)

```
<?php
/*
* The URL for SMS Central's API where your HTTP POST should be sent
*/
$url = "https://my.smscentral.com.au/wrapper/sms";
/*
*  replace the value of this variable with your username
*/
$parameters['USERNAME'] = "your username";

/*
*  replace the value of this variable with your password
*/
$parameters['PASSWORD'] = "your password";

/*
* this should always be 'send' for sending a message
*/
$parameters['ACTION'] = "send";

/*
*  to send from SMS Central's pool of shared numbers, use the string value 'shared',
* otherwise, you may use  the actual dedicated number that you have for your account
*/
$parameters['ORIGINATOR'] = "shared";

/*
*  this is the mobile number of the person you want to send this message to.
* We recommend using international format (without the + sign).
```

```
* Here's an example Australian mobile, replace this with the actual mobile you want to send
to
*/
$parameters['RECIPIENT'] = "61420314421";
/*
* The reference parameter, this should always be unique string.
* This is the value that will let you match replies with the original outbound message
* Here's an example on generating a random value, though you may want to use values
that
* are of some significance to you, such as the ID of the outbound message in your database,
etc.
*/
$parameters['REFERENCE'] = rand(0,getrandmax());
$parameters['REFERENCE'] = hash('md5', $parameters['REFERENCE']);

/*
* this should always be 'send' for sending a message
*/
$parameters['MESSAGE_TEXT'] = "Hi  Homer, you have an appointment with Dr. Nick
tomorrow at 1:30pm";

/*
* Now we can send the the HTTP POST to SMS Central with all the required parameters
* We'll do it, with cURL
*/
$request = "";
foreach ($parameters as $key=>$value)
{
   $request .= $key.'='.$value.'&';}
}
rtrim($request,'&');
$ch = curl_init;
curl_setopt($ch, CURLOPT_URL, $url);// Set the URL
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);      // Return as a variable
curl_setopt($ch, CURLOPT_POST, true);                  // Set POST method
curl_setopt($ch, CURLOPT_POSTFIELDS, $request);   // Set the POST Variables
$response = curl_exec($ch);                                    // Execute the Request
curl_close($ch);              // Close the CURL handle
if ($response == "0")
```

```
{
  /* The message was sent successfully.
   * You can end here, or store in your database, etc
  */
}
else
{
  /*
  * An error message was returned. You can log this, or email it, or re-try the cURL request,
etc.
  * Check out the API Reference for a list of possible error codes and reasons
  */
}
?>
```

The above PHP example uses the cURL library. cURL is available with PHP since PHP Version 4.0.2. To learn about cURL, check out the PHP Docs at: http://php.net/manual/en/book.curl.php

So the code above will get you going to send the actual message. You should pay attention to the response to the HTTP POST as it will let you know whether your message will be delivered, or if any error has occurred (i.e. an invalid number, out of credit, etc); please check the API Reference for a list of possible error codes and reasons.

Now that you've sent the message, it's time for replies...

NEXT, LET'S GET REPLIES.

There are two ways to get replies. You can receive them sent to your server in real-time, or you can check for new replies when you're ready. If you'd like to receive replies in real-time, you can set this up easily by logging in to your SMS Central account and setting up an 'inbound SMS' service.

In case you're yet to decide whether you want replies in real-time or not, we'll run through the API code for both methods.

- Receive Replies (in Real-Time)
- Check For Replies (On-Demand)

Receive Replies (in Real-Time)

Excellent choice. It's easy to implement too, by now you're familiar with the API and the parameters, as you've used them to send an SMS, it's the same parameters that will be supplied to your server, when you receive an SMS.

Before implementing this, the one thing you should be aware of is that you'll need to set up an 'inbound SMS service' by logging in to your SMS Central account, where you can enter the URL where you'd like the replies to be forwarded to. We recommend using a secure HTTPS URL.

To receive replies in real-time, SMS Central will be sending these replies to your server with an HTTP POST.

The parameters you should be expecting in that HTTP POST are:

USERNAME
This is your SMS Central username

PASSWORD
This is your SMS Central password

ORIGINATOR
This is the number of the person who is sending the reply message.

RECIPIENT
This is the number that the reply is going to. Basically, this will be the number that you sent your message from. If you used 'shared', then the reply will be to the SMS Central shared number that was used to send your original outbound message.

REFERENCE
When SMS Central matches this reply message with the original message that you sent out, if you provided a 'REFERENCE' value for that message, we'll send you the same value in this reply; this is so that you can match up the messages on your side too.

MESSAGE_TEXT
As you'd probably be expecting, this parameter will contain a string value of the actual message text that the person replying has sent through.

Some Code

Now that you know which parameters to expect, let's run through some sample code that you can use, to receive SMS messages (replies) in real-time to your own server.

We are providing a PHP example here below, however this can be done with any programming language (we'll add more and more code samples with different languages over time, or let us know if you need some help!)

```
<?php

$username = $_POST['USERNAME'];
$password = $_POST['PASSWORD'];
$originator = $_POST['ORIGINATOR'];
$recipient = $_POST['RECIPIENT];
$reference = $_POST['REFERENCE'];
$messagetext = $_POST['MESSAGE_TEXT'];

/* Always output a '0' for SMS Central. This is required so that SMS Central's server knows
that
*  you have received the message. If you don't output this, SMS Central's server will continue
*  trying to send you the message and may result in multiple inbound messages.
*/
echo '0';

/*  Check that the $username and $password values match your actual username and
password
*   for SMS Central, to prevent anyone from sending sending you fake inbound messages */

/* You can check the $reference value against your database, to find the corresponding
message
*  that you actually sent to this person who has replied */

/* If you expected a response, you can parse the $messagetext value. For example, if you
want to
*  check if the reply is a 'Yes' or a 'No', you can use the following code */
If (preg_match("/^yes$/i", $messagetext)
```

```
{
    // Reply was s a Yes. You can perform an action based on this.
}
else If (preg_match("/^yes$/i", $messagetext)
{
    // Reply was s a No. You can perform an action based on this.
}


?>
```

Essentially, that's all the code you need in order to receive messages.

You're now all set to send reminder & notifications messages, and receive real-time replies!

Receive Replies (On-Demand)

There are a few reasons you might not want real-time replies. Here are a a few:

Due to security reasons you don't allow any server to POST to you, or your server is not available all the time, or you may not be willing to take on the potential load that real-time replies might have on your server.

So, we let you check for replies when you're good and ready, instead. It's simple too.
It's all explained on the API Reference document, though we'll run through it here for you to get you going.

It all begins with a HTTP POST to the SMS Central API URL. Since this is just one single message, here are the required parameters (as described by the API Reference) that will apply:

USERNAME
This is your SMS Central username

PASSWORD
This is your SMS Central password

ACTION
As you are checking for received message, the value for this parameter will always be 'read'

DATESTART
You can determine how far back you want to check for messages, by providing a date value (in Datetime format, i.e. 2011-12-25 00:00:00) for this parameter.

DATEEND
You can determine the cut off date for how recent messages can be, by providing a date value (in Datetime format, i.e. 2011-01-24 23:59:59) for this parameter.

STATUS
You can determine whether you want to check for all messages, or only unread messages, or messages marked as read, with the values 'ALL', 'UNREAD' or 'READ', respectively.

So, given these parameters, let's run through the example (with code) where you only want to check for unread messages that you received yesterday.

Some code

The following code example will show you how you can check for 'unread' messages that were received 'yesterday'. You can change the parameter values to check for different times and different message status'.

The code example we are providing is in PHP, however this can be done with any programming language (we'll add more and more code samples with different languages over time, or let us know if you need some help!)

```
<?php

/*
* The URL for SMS Central's API where your HTTP POST should be sent
*/
$url = "https://extranet.smscentral.com.au/wrapper/sms";

/*
*  replace the value of this variable with your username
*/
$parameters['USERNAME'] = "your username";

/*
*  replace the value of this variable with your password
*/
$parameters['PASSWORD'] = "your password";

/*
* this should always be 'send' for sending a message
*/
$parameters['ACTION'] = "read";

/*
*  Set the Datestart parameter to the start of yesterday
*/
$parameters['DATESTART] = date("Y-m-d 00:00:00", strtotime("-1 day");

/*
```

```
*  Set the Datestart parameter to the end of yesterday
*/
$parameters['DATEEND'] = date("Y-m-d 23:59:59", strtotime("-1 day");

/*
*  We are checking for 'unread' messages only.
*/
$parameters['STATUS'] = "UNREAD";

/*
* Now we can send the the HTTP POST to SMS Central with all the required parameters
* We'll do it, with cURL
*/

$request = "";
foreach ($parameters as $key=>$value)
{
   $request .= $key.'='.$value.'&';}
}
rtrim($request,'&');

$ch = curl_init;
curl_setopt($ch, CURLOPT_URL, $url);// Set the URL
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);      // Return as a variable
curl_setopt($ch, CURLOPT_POST, true);                 // Set POST method
curl_setopt($ch, CURLOPT_POSTFIELDS, $request);   // Set the POST Variables
$response = curl_exec($ch);                                    // Execute the Request
curl_close($ch);            // Close the CURL handle

/*
*the $response variable should now contain the response with all the unread messages
* This will be in XML format, check out the next part of this guide to help you through this
*/

?>
```

So the above code will send the HTTP POST through to SMS Central, requesting all the 'unread' received messages for 'yesterday'.

Note: Once you have requested these messages, they will be marked as 'read'. So if you want to retrieve them again, you'll need to use the 'READ' value for the 'STATUS' parameter.

The response you will receive...

If there are messages, you will get an XML Response. It would be in the following format (here's an example):

```
<messages>
  <message>
    <datestamp>2011-12-26 15:14:13</datestamp>
    <direction>MO</direction>
    <originator>61xxxxxxxxx</originator>
    <recipient>61xxxxxxxxx</originator>
    <messagetext>Doh! Hi Dr. Nick, it's Homer here, can I re-schedule?</messagetext>
  </message>
  <message>
    <datestamp>2011-12-27 11:10:59</datestamp>
    <direction>MO</direction>
    <originator>61xxxxxxxxx</originator>
    <recipient>61xxxxxxxxx</originator>
    <messagetext>Hi Dr. Nick, it's Bart, I need to come in for a check-
up</messagetext>
  </message>
</messages>
```

What you'll notice in the response, is that the <messages> tag is a container holding a collection of <message> tags. Each <messagetag> and the elements within each <message> tag, contain the details of the individual message.

Check out the API reference for other possible responses, such as the case where there is an error, or no messages found.

If you are using the PHP programming language for your development, you might use the SimpleXML library to parse this XML response.

Here's a PHP code example of how to get the data into variables.

In the example below, we'll just get the text of the first message into a variable and output it to the screen.

This can be done with just about any programming language that has XML parsing libraries or capability (we'll add more and more code samples with different languages over time, or let us know if you need some help!)

The following code is continued on from the previous code sample within this 'Checking for replies (On-Demand) section'

```
//... (continued from the above code sample, you may delete this line)

/*

*the $response variable should now contain the response with all the unread messages

* This will be in XML format, check out the next part of this guide to help you through this

*/

$messages = new SimpleMLElement($response);

/*

* Let's just loop through all the messages received, just for fun.

* You can ignore this whole 'foreach' block. We've only added it here to provide an example

*/

foreach ($messages->message as $message)

{

    /*

    * You have the $message variable availale.

    * You can access each element, such as the' messagetext' with

    * $messagetext = $message->messagetext;
```

```
    */
}
/*
*Let's just get the message text from the first message, directly, and output to the screen.
*/
$firstMessageText = $messages->message[0]->messagetext;
echo $firstMessageText;
?>
```

Essentially, that's all you need in order to check for received messages.

You're now all set to send reminder & notifications messages, and check for replies on-demand!

Send a notification without replies

Easy!

It's exactly the same as sending reminder & notifications messages where you do want replies, except this time you're also able to send the message from a Name, rather than a shared or dedicated number! This means the message can appear to come from your own name, or company name.

So instead of having the value "shared", or an actual number in the "ORIGINATOR" parameter, you can use an alphanumeric name (no numbers and letters only, no spaces, max 11 characters).

This would mean your message would appear to come from the name you determine as the 'ORIGINATOR'.

In this guide, we'll provide a PHP code example of sending a message that will appear to come from "DrNick".

SO, LET'S SEND THE MESSAGE...

The first thing you'll need to do is send the message, via a HTTP POST to the SMS Central API URL. Since this is just one single message, here are the required parameters (as described by the API Reference) that will apply:

USERNAME
This is your SMS Central username

PASSWORD
This is your SMS Central password

ACTION
As you are sending a message, the value for this parameter will always be 'send'

ORIGINATOR
You can specify an alphanumeric value (such as a name, or company name, etc) containing numbers and letters only, up to a maximum of 11 characters. Your message will appear to come from this name, rather than a number.

RECIPIENT
This is the number (in international format, i.e. 61420314421 for an Australian mobile) that you want to send the message to.

REFERENCE
You can supply a unique (must be unique) reference value which is relevant to you, and you'll get the same reference value with any reply (so that you can match the reply to the original sent message)

MESSAGE_TEXT
You'd probably have guessed this one, this is the parameter that contains the text of your message. Remember, a message is 160 characters in length, any longer than that and it would be sent as 2 messages, or 3 messages, etc.

Some code

Now that you know what you'd need to send, let's delve into the code itself and how you would do it.

We are providing a PHP example here below, however this can be done with any programming language (we'll add more and more code samples with different languages over time, or let us know if you need some help!)

```
<?php
/*
* The URL for SMS Central's API where your HTTP POST should be sent
*/
$url = "https://extranet.smscentral.com.au/wrapper/sms";
/*
*  replace the value of this variable with your username
*/
$parameters['USERNAME'] = "your username";
/*
*  replace the value of this variable with your password
*/
$parameters['PASSWORD'] = "your password";
/*
* this should always be 'send' for sending a message
*/
$parameters['ACTION'] = "send";
/*
*  to send from SMS Central's pool of shared numbers, use the string value 'shared',
* otherwise, you may use  the actual dedicated number that you have for your account
*/
$parameters['ORIGINATOR'] = "DrNick";
/*
*  this is the mobile number of the person you want to send this message to.
* We recommend using international format (without the + sign).
* Here's an example Australian mobile, replace this with the actual mobile you want to send
to
*/
$parameters['RECIPIENT'] = "61420314421";
```

```
/*
* The reference parameter, this should always be unique string.
* This is the value that will let you match replies with the original outbound message
* Here's an example on generating a random value, though you may want to use values
that
* are of some significance to you, such as the ID of the outbound message in your database,
etc.
*/
$parameters['REFERENCE'] = rand(0,getrandmax());
$parameters['REFERENCE'] = hash('md5', $parameters['REFERENCE']);
/*
* this should always be 'send' for sending a message
*/
$parameters['MESSAGE_TEXT'] = "Hi  Homer, just a reminder that we have an
appointment tomorrow at 1:30pm";
/*
* Now we can send the the HTTP POST to SMS Central with all the required parameters
* We'll do it, with cURL
*/
$request = "";
foreach ($parameters as $key=>$value)
{
  $request .= $key.'='.$value.'&';}
}
rtrim($request,'&');
$ch = curl_init;
curl_setopt($ch, CURLOPT_URL, $url);// Set the URL
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);     // Return as a variable
curl_setopt($ch, CURLOPT_POST, true);               // Set POST method
curl_setopt($ch, CURLOPT_POSTFIELDS, $request);  // Set the POST Variables
$response = curl_exec($ch);                                  // Execute the Request
curl_close($ch);           // Close the CURL handle
if ($response == "0")
{
  /* The message was sent successfully.
   * You can end here, or store in your database, etc
  */
}
```

```
else
{
  /*
  * An error message was returned. You can log this, or email it, or re-try the cURL request,
etc.
  * Check out the API Reference for a list of possible error codes and reasons
  */
}
?>
```

The above PHP example uses the cURL library. cURL is available with PHP since PHP Version 4.0.2. To learn about cURL, check out the PHP Docs at: http://php.net/manual/en/book.curl.php

So the code above will get you going to send the actual message. You should pay attention to the response to the HTTP POST as it will let you know whether your message will be delivered, or if any error has occurred (i.e. an invalid number, out of credit, etc); please check the API Reference for a list of possible error codes and reasons.

That's really all there is to it. You're now set to sent SMS Notifications and Reminders.